Chapter 1

R Preliminaries

1.1 Matrix Defined, Deeper Understanding Using Software

Scientists and accountants often use a set of numbers arranged in rectangular arrays. Scientists and mathematicians call rectangular arrays matrices. For example, a 2×3 matrix of six numbers is defined as:

$$A = \{a_{ij}\} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} = \begin{bmatrix} 3 & 5 & 11 \\ 6 & 9 & 12 \end{bmatrix}, \tag{1.1}$$

where the first subscript $i = 1, 2$ of a_{ij} refers to the row number and the second subscript $j = 1, 2, 3$ refers to the column number.

Such matrices and their generalizations will be studied in this book in considerable detail. Rectangular arrays of numbers are called spreadsheets or workbooks in accounting parlance and 'Excel' software has standardized them in recent decades. The view of rectangular arrays of numbers by different professions can be unified. The aim of this book is to provide tools for developing a deeper understanding of the reality behind rectangular arrays of numbers by applying many powerful results of matrix algebra along with certain software and graphics tools.

Exercise 1.1.1: Construct a 2×2 matrix with elements $a_{ij} = i + j$. [Hint: the first row will have numbers 2,3 and second row will have 3,4].

Exercise 1.1.2: Given integers 1 to 9 construct a 3×3 matrix with first row having numbers 1 to 3, second row with numbers 4 to 6 and last row with numbers 7 to 9.

The R software allows a unified treatment of all rectangular arrays as 'data frame' objects. Although matrix algebra theorems do not directly apply to data frames, they are a useful preliminary, allowing matrix algebra to be a very practical and useful tool in everyday life, not some esoteric subject for scientists and mathematicians. For one thing, the data frame

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

objects in R software allow us to name the rows and columns of matrices in a meaningful way. We can, of course, strip away the row-column names when treating them as matrices. Matrix algebra is abstract in the sense that its theorems hold true irrespective of row-column names. However, the deeper understanding of the reality behind those rectangular arrays of numbers requires us to have easy access to those names when we want to interpret the meaning of matrix algebra based conclusions.

1.2 Introduction, Why R?

The New York Times, 6 January 2009 had an article by Daryl Pregibon, a research scientist at *Google* entitled "Data Analysts Captivated by R's Power." It said that the software and graphics system called R is fast becoming the *lingua franca* of a growing number of data analysts inside corporations and academia. Countless statisticians, engineers and scientists without computer programming skills find R "easy to use." It is hard to believe that R is free. The article also stated that "R is really important to the point that it's hard to overvalue it."

R is numerically one of the most accurate languages, perhaps because it is free with transparent code which can be checked for accuracy by almost anyone, anywhere. R is supported by volunteer experts from around the world and available anywhere one has access to (high speed) Internet. R works equally well on Windows, Linux, and Mac OS X computers. R is based on an earlier public domain language called S developed in Bell Labs in 1970's. I had personally used S when I was employed at Bell Laboratories. At that time it was available only on UNIX operating system computers based on principles of object oriented languages. S-PLUS is the commercial version of S, whereas R is the free version. It is a very flexible object oriented language (OOL) in the sense that inputs, data and outputs are all objects (e.g., files) inside the computer.

R is powerful and fun: R may seem daunting at first to someone who has never thought of himself or herself as a programmer. Just as anyone can use a calculator without being a programmer, anyone can start using R as a calculator. For example, when balancing a checkbook R can help by adding a bunch of numbers '300+25.25+33.26+12*6', where the repeated number 12 need not be typed six times into R. The advantage with R over a calculator is that one can see the numbers being added and correct any errors on the go. The tip numbered xxvi among the tips listed in Sec. 1.5

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

shows how to get the birth day-of-the-week from a birthday of a friend. It can also give you the date and day-of-the-week 100 days before today and fun things like that. There are many data sets already loaded in R. The dataset 'women' contains Average Heights and Weights for American Women. The simple R command 'summary(women)' gives basic descriptive statistics for the data (minimum, maximum, quartiles, median and mean). No calculator can do this so conveniently. The dataset named 'co2' has Mauna Loa Atmospheric CO2 Concentration to check global warming. The reader will soon discover that it is more fun to use R rather than any calculator.

Actually, easy and powerful graphics capabilities of R make it fun for many of my students. The command 'example(points)' gives code examples showing all kinds of lines along points and creation of sophisticated symbols and shapes in R graphs. A great Internet site for all kinds of R graphics will convince anyone how much fun one can have with R: `http://AddictedToR.free.fr/graphiques/index.php`

The reason why R a very powerful and useful calculator is that thousands of 'functions' and program packages are already written in R. The packages are well documented in standard format and have illustrative examples and user-friendly vignettes. The user simply has to know what the functions do and go ahead and use them *at will*, for free. Even if R is a "language" it is an "interpreted" language similar to a calculator, and the language C, not "compiled" language similar to FORTRAN, GAUSS, or similar older languages.

Similar to a calculator, all R commands are implemented as they are received by R (typed). Instead of having subroutines similar to FORTRAN, R has 'functions.' Calculations requiring hundreds of steps can be defined as 'functions' and subsequently implemented by providing values for the suitable number of arguments to these functions. A typical R package has several functions and data sets. R is functional and interpreted computer language which can readily import and employ the code for functions written in other languages including C, C++, FORTRAN, among others.

John M. Chambers, one of my colleagues at Bell Labs, is credited with creating S and helping in the creation of R. He has published an article entitled "The Future of R" in the first issue (May 2009) of the 'R Journal' available on the Internet at:

`http://journal.r-project.org/\break2009-1/RJournal\_2009-1\_Chambers.pdf`

John lists six facets of R

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

(i) an interface to computational procedures of many kinds;
(ii) interactive, hands-on in real time;
(iii) functional in its model of programming;
(iv) object-oriented, "everything is an object";
(v) modular, built from standardized pieces; and,
(vi) collaborative, a world-wide, open-source effort.

This list beautifully explains the enormous growth and power of R in recent years. Chambers goes on to explain how R functions are themselves objects with their own functionality.

I ask my students to first type a series of commands into a text editor and then copy and paste them into R. I recommend the text editor called Tinn-R, freely available at: (`http://www.sciviews.org/Tinn-R/`). The reader should go to the bottom of the web page and download it to his or her computer. Next, click on "Setup for Tinn-R" Be sure to use the old stable version (1.17.2.4) (.exe, 5.2 Mb) compatible with Rgui in SDI or MDI mode. Tinn-R color codes the R commands and provides helpful hints regarding R syntax in the left column entitled 'R-card.' Microsoft Word is also a good text editor to use, but care is needed to avoid smart or slanted quotation marks (unknown to R).

1.3 Obtaining R

One can *Google* the word r-project and get the correct Internet address. The entire software is 'mirrored' or repeated at several sites on the Internet around the world. One can go to

(`http://www.r-project.org`) and choose a geographically nearby mirror. For example, US users can choose

`http://cran.us.r-project.org`

Once arriving at the appropriate Internet site, one looks on the left hand side under "Download." Next, click on CRAN Link. Sometimes the site asks you to pick a mirror closest to your location again. Click on 'Windows' (if you have a Windows computer) and then Click on 'base.'

For example, if you are using Windows XP computer and if the latest version of R is 2.9.1 then click on "Download R-2.9.1 for Windows". It is a good idea to download the setup program file to a temporary directory which usually takes about 7-8 minutes if you have a reasonably fast connection to the Internet. Now, double click on the 'setup' file (R-2.9.1-win32.exe) to get R set up on any local computer. When the setup wizard

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

will first ask you language, choose English. Then click 'Next' and follow other on-screen instructions.

If you have a Windows Vista computer, among frequently asked question, they have answer to the question: How do I install R when using Windows Vista? One of the hints is 'Run R with Administrator privileges in sessions where you want to install packages.'

Choose simple options. Do not customize it. The setup does everything, including creating an Icon for R-Gui (graphical user interface). It immediately lets you know what version is being used.

1.4 Reference Manuals in R

Starting at the R website (`http://cran.r-project.org`) left column click on 'manuals' and access the first bullet point called "An Introduction to R," which is the basic R manual with about 100 pages. It can be browsed in html format at:

`http://cran.r-project.org/doc/manuals/R-intro.html`

It can also be downloaded in the pdf format by clicking at a link within that bullet as:

`http://cran.r-project.org/doc/manuals/R-intro.pdf`

The reader is encouraged to read all chapters, especially Chapter 5 of 'R-intro' dealing with arrays and Chapter 12 dealing with graphics.

There are dozens of free books available on the Internet about R at:

`http://cran.r-project.org/doc/contrib`

The books are classified into 12 books having greater than 100 pages starting with "Using R for Data Analysis and Graphics - Introduction, Examples and Commentary" by John Maindonald. Section 7.8 of the above pdf file deals specifically with matrices and arrays in R.

`http://cran.r-project.org/doc/contrib/usingR.pdf`

There are additional links to some 18 (as of July 2009) short books (having less than 100 pages). The list starts with "R for Beginners" by Emmanuel Paradis.

Social scientists and economists have found my own book Vinod (2008a) containing several examples of R code for various tasks a useful reference. The website of the book also contains several solved problems among its exercises. A website of a June 2009 conference on R in social sciences at Fordham University in New York is: `http://www.cis.fordham.edu/QR2009`. The reader can view various presentations by distinguished researchers who

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

use R.

1.5 Basic R Language Tips

Of course, the direct method of learning any language is to start using it. I recommend learning the assignment symbols '=' or '<-' and the combine symbol 'c' and start using R as a calculator. A beginning user of R should consult one or more of the freely available R manuals mentioned in Sec. 1.4. Nevertheless, this section lists some practical points useful to beginning users, as well as, some experienced users of R.

(i) Unlike paper and pencil mathematics, (*) means multiply; either the hat symbol ($\wedge$) or ($**$) are used for raising to a power; (|) means the logical 'or' and (!) means logical negation.
(ii) If a command is inside simple parentheses (not brackets) it is printed to the screen.
(iii) Colon (:) means generate a sequence. For example, x=3:7 creates an R (list or vector) object named 'x' containing the numbers (3,4,5,6,7).
(iv) Generally speaking, R ignores all spaces inside command lines. For example, 'x=1:3' is the same as 'x = 1 : 3'.
(v) Comments within R code are put anywhere, starting with a hash-mark ('#'), such that everything from that point to the end of the line is a comment by the author of the code (usually to oneself) and completely ignored by the R processor.
(vi) 'c' is a very important and very basic function in R which combines its arguments. One cannot really use R without learning to use the 'c' function. For example, 'x=c(1:5, 17, 99)' will create a 'vector' or a 'list' object called 'x' with elements: (1, 2, 3, 4, 5, 17, 99). The 'c' stands for 'concatenate', 'combine' or 'catalog.'
(vii) Generally, each command should be on a separate line of code. If one wants to type or have two or more commands on one line, a semi-colon (;) must be used to separate them.
(viii) R is **case sensitive** similar to UNIX. For example, the lower case named object b is different from upper case named object B.
(ix) The object names usually start with a letter, never with numbers. Special characters including underscores are not allowed in object names but periods (.) are allowed.
(x) R may be used as a calculator. The usual arithmetic operators +, -,

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

*, / and the hat symbol (∧) for raising to a power are available. R also has log, exp, sin, cos, tan, sqrt, min, max with self-explanatory meanings.

(xi) Expressions and assignments are distinct in R. For example, if you type the expression:

```
2+3+4^2
```

R will proceed to evaluate it. If you assign the expression a name 'x' by typing:

```
x=2+3+4^2; x
```

The resulting object called 'x' will contain the evaluation of the expression and will be available under the name 'x,' but not automatically printed to the screen. Assuming that one wants to see R print it, one needs to type 'x' or 'print(x)' as a separate command. We have included 'x' after the semi-colon (;) to suggest a new command asking R to print the result to the screen.

(xii) The R language and its packages together have tens of thousands of functions for doing certain tasks on their arguments. For example, 'sqrt' is a function in R which computes the square root of its argument. It is important to use simple parentheses as in 'print(x)' or 'sqrt(x)' when specifying the arguments of R functions. Using curly braces or brackets instead of parentheses will give syntax error (or worse confusion) in R.

(xiii) Brackets are used to extract elements of an array. For example, 'x[2]' extracts second element of a list 'x'. Similarly, 'x[2,1]' extracts the number along the second row and first column of a matrix x. Again, using curly braces or parentheses will not work for this purpose.

(xiv) Curly braces ('{' and '}') are used to combine several expressions into one procedure.

(xv) Re-executing previous commands is done by using vertical arrow keys on the keyboard. Modifications to such commands can be made by using horizontal arrows on the keyboard.

(xvi) Typing 'history()' provides a list of all recent commands typed by the user. These commands can then be copied into MS-Word or any text editor, suitably changed, copied and pasted back into R.

(xvii) R provides numerical summaries of data. For example, let us use the vector of seven numerical values mentioned above as 'x' and let us issue certain commands in R (especially the summary command) collected in the following R software input snippet.

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

```
#R.snippet
x=c(1:5, 17, 99)
summary(x)
```

Then R will evaluate them and report to the screen the following output.

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1.00    2.50    4.00   18.71   11.00   99.00
```

Note that the function 'summary' computes the minimum, first quartile (25% of data are below this and 75% are above this), median, third quartile (75% of data are below this and 25% are above this), and the maximum.

(xviii) Attributes (function 'attr') can be used to create a matrix. The following section discusses the notion of 'attributes' in greater detail. It is particularly useful for matrix algebra.

```
#R.snippet
x=c(1:4, 17, 99) # this x has 6 elements
attr(x,"dim")=c(3,2) #this makes x a 3 by 2 matrix
x
```

Then R will place the elements in the object x column-wise into a 3 by 2 matrix and report:

```
     [,1] [,2]
[1,]    1    4
[2,]    2   17
[3,]    3   99
```

(xix) R has extensive graphics capabilities. The plot command is very powerful and will be illustrated at several places in this book.

(xx) R can by tailor-made for specific analyses.

(xxi) R is an Interactive Programming language, but a set of R programs can be submitted in a batch mode also.

(xxii) R distinguishes between the following 'classes' of objects: "numeric", "logical", "character." "list", "matrix", "array", "factor" and "data.frame." The 'summary' function is clever enough to do the right summarizing upon taking into account the class.

(xxiii) It is possible to convert objects from one 'class' to another within limits. Let us illustrate this idea. Given a vector of numbers, we can convert into a logical vector containing only two items TRUE and

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

FALSE. by using the function 'as.logical'. All numbers (positive or negative) are made TRUE and the number zero is made FALSE. Conversely, given a logical vector, we can convert it into numerical vector by using the function 'as.numeric.' The reader should see that this converts TRUE to the number 1 and FALSE to the number 0. The R function 'as.complex' converts the numbers into complex numbers with the appropriate coefficient (=0) for the imaginary part of the complex number identified by the symbol $i = \sqrt{(-1)}$. The 'as.character' function places quotes around the elements of a vector.

R program snippet **1.5.1** (as.something) is next.

```
x=c(0:3, 17, 99) # this x has 6 elements
y=as.logical(x);y#makes 0=FALSE all numbers=TRUE
as.numeric(y)#evaluate as 1 or 0
as.complex(x)#will insert i
as.character(x) #will insert quotes
```

The following illustrates how R converts between classes using the 'as.something' command. The opeartion of 'as.matrix' and 'as.array' will be illustrated in the following Sec. 1.7.

```
> y=as.logical(x);y#
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> x=c(0:3, 17, 99) # this x has 6 elements
> y=as.logical(x);y#makes 0=FALSE all numbers=TRUE
[1] FALSE  TRUE  TRUE  TRUE  TRUE  TRUE
> as.numeric(y)#evaluate as 1 or 0
[1] 0 1 1 1 1 1
> as.complex(x)#will insert i
[1]  0+0i  1+0i  2+0i  3+0i 17+0i 99+0i
> as.character(x) #will insert quotes
[1] "0"  "1"  "2"  "3"  "17" "99"
```

(xxiv) Missing data in R are denoted by "NA" without the quotation marks. These are handled correctly by R. In the following code we deliberately make the x value along third row and second column as NA, that is we specify that x[3,2] as missing. The exclamation symbol (!) is used as a logical 'not'. Hence if 'is.na' means is missing, '!is.na' means 'is not missing'. The bracketed conditions in R with repetition of 'x' outside and inside the brackets are very useful for subset selection. For example, x[!is.na(x)] and x[is.na(x)] in the following snippet cre-

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

ate subsets to x containing only the non-missing values and only the missing values, respectively.

R program snippet **1.5.2** is next.

```
x=c(1:4, 17, 99) # this x has 6 elements
attr(x,"dim")=c(3,2) #this makes x a 3 by 2 matrix
x[3,2]=NA
y=x[!is.na(x)] #picks only non-missing subset of x
z=x[is.na(x)] #picks only the missing subset of x
summary(x);summary(y);summary(z)
```

Then R will recognize that 'x' is a 3 by 2 matrix and the 'summary' function in the snippet 1.5.2 will compute the summary statistics for the two columns separately. Note that 'y' converts 'x' matrix into an array of 'non-missing' set of values and computes their summary statistics. The vector 'z' contains only the one missing value in the location at row 3 and column 2. Its summary (under column for second variable 'V2' below correctly recognizes that one item is missing.

```
> summary(x);
       V1              V2
 Min.   :1.0   Min.   : 4.00
 1st Qu.:1.5   1st Qu.: 7.25
 Median :2.0   Median :10.50
 Mean   :2.0   Mean   :10.50
 3rd Qu.:2.5   3rd Qu.:13.75
 Max.   :3.0   Max.   :17.00
               NA's   : 1.00

> summary(y)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    1.0     2.0     3.0     5.4     4.0    17.0

> summary(z)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
                                                    1
```

Recall that 'y' picks out only the non-missing values, whereas z picks out the one missing value denoted by 'NA'. Note also that R does not foolishly try to find the descriptive statistics of the missing value.

(xxv) R admits character vectors. For example,

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

```
#R.snippet
x=c("In", "God", "We",  "Trust", ".")
x; print(x, quote=FALSE)
```

We need the print command to have the options 'quote=FALSE' to prevent printing of all individual quotation marks as in the output below.

```
[1] "In"    "God"   "We"    "Trust" "."
[1] In    God   We    Trust .
```

Character vectors are distinguished by quotation marks in R. One can combine character vectors with numbers without gaps by using the 'paste' function of R and the argument separation with a blank as in the following snippet.

```
#R.snippet
paste(x,1:4, sep="")
paste(x,1:4, sep=".")
```

Note that 'x' contains five elements including the period (.). If we attach the sequence of only 4 numbers, R is smart and recycles the numbers 1:4 again and again till needed. The argument '(sep="")' is designed to remove any space between them. If sep=".", R wil place the (.) in between. Almost anything can be placed between the characters by using the 'sep' option.

```
[1] "In1"    "God2"   "We3"    "Trust4" ".1"
> paste(x,1:4, sep=".")
[1] "In.1"    "God.2"   "We.3"    "Trust.4" "..1"
```

(xxvi) Date objects. Current date including time is obtained by the command 'date(),' where the empty parentheses are needed.

```
date()
[1] "Sat Jul 11 14:24:38 2009"
Sys.Date()
weekdays(Sys.Date()-100) #plural weekday
weekdays(as.Date("1971-09-03"))
#find day-of-week from birthdate
```

The output of the above commands is given below. Note that only the system date (without the time) is obtained by the command 'Sys.Date()'. It is given in the international format as YYYY-mm-dd for year, month and day.

Find the date 30 days from the current date:

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

If you want to have the date after 30 days use the command 'Sys.Date()+30'.

Find the day of week 100 days before the current date:
If you want to know the day of week 100 days before today, use the R command 'weekdays(Sys.Date()-100).' If you want to know the day of the week from a birth-date, which is on September 3, 1971, then use the command 'weekdays(as.Date("1971-09-03"))' with quotes as shown, and R will respond with the correct day of the week: Friday.

```
> date()
[1] "Sat Jul 11 20:04:41 2009"
> Sys.Date()
[1] "2009-07-11"
> weekdays(Sys.Date()-100) #plural weekday
[1] "Thursday"
> weekdays(as.Date("1971-09-03"))
#find day-of-week from birth-date
[1] "Friday"
```

These are some of the fun uses of R. An R package called 'chron', James and Hornik (2010), allows further manipulation of all kinds of date objects.

(xxvii) Complex number $i = \sqrt{(-1)}$ is denoted by '1i'. For example, if x=1+2i and y=1-2i, the product $xy = 1^2 + 2^2 = 5$ as seen in the following snippet

R program snippet **1.5.3** (dot product) is next.

```
x=1+2*1i  #number 1 followed by letter i is imaginary i in R
y=1-2*1i
x*y #dot product of two imaginary numbers
```

```
> x*y #dot product of two imaginary numbers
[1] 5+0i
```

1.6 Packages within R

Recall from Sec. 1.3 that John Chambers has noted that R is modular and collaborative. The 'modules' mostly contain so-called 'functions' in R, which are blocks of R code which follow some rules of syntax. Blocks of input code submitted to R are called snippets in this book. These can call existing functions and do all kinds of tasks including reading and writing

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

of data, writing of citations, data manipulations and plotting.

Typical R functions have some 'input' information or objects, which are then converted into some 'output' information or objects. The rules of syntax for creation of functions in R are specifically designed to facilitate collaboration and extension. Since R is 'open source,' the underlying code (however complicated) for all R functions is readily available for extension and /or modification. For example, typing 'matrix' prints the internal R code for the R function matrix to the screen, ready for modification. Of course, more useful idea is to type '?matrix' (no spaces, lower case m) to get the relevant user's manual page for using the function 'matrix'.

The true power of R can be exploited if one knows some basic rules of syntax for R functions, even if one may never write a new function. After all, hundreds of functions are already available for most numerical tasks in basic R and thousands more are available in various R packages. It is useful to know (i) how to create an object containing the output of a funtion function already in the memory of R, and (ii) the syntax for accessing the outputs of the function by the dollar symbol suffix. Hence, I illustrate these basic rules using a rather trivial R function for illustration purposes.

R snippet **1.6.1** explains inputs /outputs of R functions.

```
myfunction=function(inp1,inp2, verbos=FALSE)
{ #function code begins with a curly brace
if(verbos) {print(inp1); print(inp2)}
out1=(inp1)^2 #first output squares the first input
out2=sin(inp2)#second output computes sin of second input
if(verbos) {print(out1); print(out2)}
list(out1=out1, out2=out2)#syntax for listing outputs
} #function code ENDS with a curly brace
```

The reader should copy and paste all the lines of snippet 1.6.1 into R and wait for the R prompt. Only if the function is logically consistent, R will return the prompt. If some errors are present, R will try to report to the screen those errors.

The first line of the function code in snippet 1.6.1 declares its name as 'myfunction' and lists the two objects called inp1 and inp2 as *required* inputs. The function also has an *optional* logical variable 'verbos=FALSE' as its third input to control the printing with a default setting (indicated by the equal symbol) as FALSE. All input objects must be listed inside parentheses (not brackets) separated by commas.

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

Since a function typically has several lines of code we must ask R to treat all of them together as a set. This is accomplished by placing them inside two curly braces. This is why the second line has left curly brace ({).

The third line of the snippet 1.6.1 has the command: 'if(verbos) print(inp1); print(inp2)'. This will optionally print the input objects to the screen if 'verbos' is 'TRUE' during a call to the function as 'myfunction(inp1,inp2, verbos=TRUE)'.

Now we enter the actual tasks of the function. Our (trivial) function squares the first input 'inp1' object and returns the squared values as the first output as an output object named 'out1'. The writer of the function has full freedom to name the objects (within reason so as not to conflict with other R names) The code "out1=inp12' creates the 'out1' for our function.

The function also needs to find the sine of second 'inp2' object and return the result as the second output named 'out2' here. This calculation is done here (still within the curly braces) by the code on the fifth line of the snippet 1.6.1. line 'out2=sin(inp2)'.

Once the outputs are created, the author of the R function must decide what names they will have as output objects. The author has the option to choose an output name different from the name internal to the function. The last but one line of a function usually has a 'list' of output objects. In our example it is 'list(out1=out1, out2=out2)'. Note that it has strange repetition of names. This is actually not strange, but allows the author complete freedom to name objects inside his function. The 'out1=out1' simply means that the author does not want to use an external name distinct from the internal name. For example, if the author had internally called the two outputs as o1 and o2, the list command would look like 'list(out1=o1, out2=o2)'. The last line of a function is usually a curly brace (}).

Any user of this function then 'calls' the R function 'myfunction' with the R command "myout=myfunction(in1,in2)". This command will not work unless the function object 'myfunction' and input objects inp1 and inp2 are already in the current memory of R. That is, the snippet 1.6.1 must be loaded and R must give hack the prompt before we can use the function.

Note that the symbol 'myout' on the left hand side of this R command is completely arbitrary and chosen by the user. Short and unique names are advisable. If no name is chosen, the command "myfunction(inp1,inp2)" will generally report certain outputs created by the function to the screen. By simply typing the chosen name 'myout' with carriage return, sends the output to the screen.

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

By choosing the name 'myout' we send the output of the function to an object bearing that name inside R. Self-explanatory names are advisable. More important, the snippet shows how to access the objects created by any existing R function for further manipulation by anyone by using the dollar symbol as a suffix followed by the name of the output object to the name of the object created by the function.

R snippet **1.6.1b** illustrates calling of R functions.

```
#Assume myfunction is already in memory of R
inp1=1:4 #define first input as 1,2,3,4
inp2=pi/2 #define second input as pi by 2
myout=myfunction(inp1,inp2)#creates object myout from myfunction
myout # reports the list output of myfunction
myout$out1^2  #^2 means raise to power 2 the output called out1
13*myout$out2#compute 13 times the second output out2
```

The snippet 1.6.1b illustrates how to create an object called 'myout' by using the R function called 'myfunction.' The dollar symbol attached to 'myout' then allows complete access to the outputs out1 and out2 created by 'myfunction.'

The snippet illustrates how to compute and print to the screen the square of the first output called 'out1'. It is accessed as an object under the name 'myout$out1,' for possible further manipulation, where the dollar symbol is a suffix. The last line of the snippet 1.6.1b shows how to compute and print to the screen 13 times second output object called 'out2' created by 'myfunction' of the snippet 1.6.1. The R output follows:

```
> myout # reports the list output of myfunction
$out1
[1]  1  4  9 16

$out2
[1] 1

> myout$out1^2  #^2 means raise to power 2 the output called out1
[1]   1  16  81 256
> 13*myout$out2#compute 13 times the second output out2
[1] 13
```

The rules of syntax in writing R functions are fairly simple and intuitive.

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

In any case, tens of thousands of (open source) functions have already been written to do almost any task spread over two thousand R packages. The reader can freely modify any of them (giving proper credit to original authors).

Accessing outputs of R functions with dollar symbol convention. The point to remember from the above illustration is that one need not write down the output of an R function on a piece of paper and type it in as an input to some other R function. R offers a simple and standardized way (by using the dollar symbol and output name as a suffix after the symbol) to cleanly access the outputs from any R function and then use them as inputs to another R function or manipulate collect, and report them as needed.

The exponential growth and popularity of R can be attributed to various convenient facilities to build on the work of others. The standardized access to function outputs by using the dollar symbol is one example of such facility in R. A second example is how R has standardized the non-trivial task of package writing. Every author of a new R package must follow certain detailed and strict rules describing all important details of data, inputs and outputs of all included functions with suitably standardized look and feel of all software manuals. Of course, important additional reasons facilitating world-wide collaboration include the fact that R is OOL, free and open source.

R packages consist of collections of such functions designed to achieve some specific tasks relevant in some scientific discipline. We have already encountered 'base' and 'contrib' packages in the process of installation of R. The 'base' package has the collection of hundreds of basic R functions belonging to the package. The "contrib" packages have other packages and add-ons. They are all available free and on demand as follows. Use the R-Gui menu called 'Packages' and from the menu choose 'install packages.' R asks you to choose a local comprehensive R archive network (CRAN) mirror or "CRAN mirror". Then R lets you "Install Packages" chosen by name from the long alphabetical list of some two thousand names. The great flexibility and power of R arises from the availability of these packages.

The users of packages are requested to give credit to the creators of packages by citing the authors in all publications. After all the developers of free R packages do not get any financial reward for their efforts. The 'citation' function of the 'base' package of R reports to the screen detailed citation information about citing any package.

Since there are over two thousand R packages available at the R website,

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

it is not possible to summarize what they do. Some R packages of interest to me are described at my web page at:

`http://www.fordham.edu/economics/vinod/r-lang.doc`

My students find this file containing my personal notes about R (MS Word file some 88 pages long) as a time-saving devise. They use the search facility of MS Word on it.

One can use the standard *Google* search to get answers to R queries, but this can be inefficient since it catches the letter R from anywhere in the document. Instead, I find that it is much more efficient to search answers to any and all R-related questions at:

`http://www.rseek.org/`

1.7 R Object Types and Their Attributes

R is OOL with various types of objects. We note some types of objects in this section. (i) vector objects created by combining numbers separated by commas are already encountered above.

(ii) 'matrices' are objects in R which generalize vectors to have two dimensions. The 'array' objects in R can be 3-dimensional matrices.

(iii) 'factors' are categorical data objects.

Unlike some languages, R allows object to be mixed type with some vectors numerical and other vectors with characters. For example in stock market data, the Ticker symbol is a character vector and stock price is a numerical vector.

(iv) 'Complex number objects.' In mathematics numerical vectors can be complex numbers with $i = \sqrt{(-1)}$ attached to the imaginary part.

(v) R objects called 'lists' permit mixtures of all these types of objects.

(vi) Logical vectors contain only two values: TRUE and FALSE. R does understand the abbreviation T and F for them.

How does R distinguish between different types of objects? Each object has certain distinguishing 'attributes' such as 'mode' and 'length.' It is interesting that R can convert some objects by converting their attributes. For example, R can convert the numbers 1 and 0 into logical values TRUE and FALSE and vice versa. The 'as.logical' function of R applied to a bunch of zeros and ones makes them a corresponding bunch of TRUE and FALSE values. This is illustrated in Sec. 1.7.1

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

1.7.1 *Dataframe Matrix and Its Summary*

We think of matrix algebra in this book as a study of some relevant information. A part of the information can be studied in the form of a matrix. The 'summary' function is a very fundamental tool in R.

For example, most people have to contend with stock market or financial data at some point in their lives. Such data are best reported as 'data frame' objects in R, having matrix-like structure to deal with any kind of 'data matrices.' In medical data the sex of the patient is a categorical vector variable, patient name is a 'character' vector whereas patient pulse is a numerical variable. All such variables can be handled satisfactorily in R. The matrix object in R is reserved for numerical data. However, many operations analogous to those on numerical matrices (e.g., summarize the data) can be suitably redefined for non-numerical objects. We can conveniently implement all such operations in R by using the concept of data frame objects invented in the S language which preceded R.

We illustrate the construction of a data frame object in R by using the function 'data.frame' and then using the 'summary' function on the data frame object called 'mydata'. Note that R does a sensible job of summarizing the mixed data frame object containing character, numerical, categorical and logical variables. It also uses the 'modulo' function which finds the remainder of division of one number by another.

R program snippet **1.7.1.1** is next.

```
x=c("In", "God", "We",  "Trust", ".")
x; print(x, quote=FALSE)
y=1:5 #first 5 numbers
z=y%%2  #modulo division of numbers by 2
# this yields 1,0,1,0,1 as five numbers
a=as.logical(z)
zf=factor(z)#create a categorical variable
data.frame(x,y,z,a, zf)
#rename data.frame as object mydata
mydata=data.frame(x,y,z,a, zf)
summary(mydata)
length(mydata) #how many rows?
```

```
> data.frame(x,y,z,a, zf)
      x y z     a zf
1    In 1 1  TRUE  1
```

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

```
2   God 2 0 FALSE  0
3    We 3 1  TRUE  1
4 Trust 4 0 FALSE  0
5     . 5 1  TRUE  1
> mydata=data.frame(x,y,z,a, zf)
> summary(mydata)
     x           y           z           a           zf
 .    :1   Min.   :1   Min.   :0.0   Mode :logical   0:2
 God  :1   1st Qu.:2   1st Qu.:0.0   FALSE:2         1:3
 In   :1   Median :3   Median :1.0   TRUE :3
 Trust:1   Mean   :3   Mean   :0.6   NA's :0
 We   :1   3rd Qu.:4   3rd Qu.:1.0
           Max.   :5   Max.   :1.0
> length(mydata)
[1] 5
```

The 'summary' function of R was encountered in the previous section (Sec. 1.5). It is particularly useful when applied to a data frame object. The 'summary' sensibly reports the number of times each word or symbol is repeated for character data, the number of True and False values for logical variables, and the number of observations in each category for categorical or factor variables.

Note that the length of the data frame object is reported as the number of rows in it. Each object comprising the data frame also has the exact same length 5. Without this property we could not have constructed our data frame object by the function 'data.frame.' If all objects are numerical R has the 'cbind' function to bind the columns together into one matrix object. If objects are mixed type with some 'character' type columns then the 'cbind' function will convert them all into 'character' type before binding the columns together into one character matrix object.

We now illustrate the use of 'as.matrix' and 'as.array' functions using the data frame object called 'mydata' inside the earlier snippet. The following snippet will not work unless the earlier snippet is in the memory of R.

R program snippet **1.7.1.2** (as.matrix, as.list) is next.

```
#R.snippet Place previous snippet into R memory
#mydata=data.frame(x,y,z,a, zf)
as.matrix(mydata)
as.list(mydata)
```

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html

Note from the following output that 'as.matrix' simply places quotes around the elements of the data frame object. The 'as.list' converts the character vector 'x' into a 'factor' arranges them alphabetically as 'levels'.

```
> as.matrix(mydata)
     x       y   z   a       zf
[1,] "In"    "1" "1" " TRUE" "1"
[2,] "God"   "2" "0" "FALSE" "0"
[3,] "We"    "3" "1" " TRUE" "1"
[4,] "Trust" "4" "0" "FALSE" "0"
[5,] "."     "5" "1" " TRUE" "1"

> as.list(mydata)
$x
[1] In    God   We    Trust .
Levels: . God In Trust We

$y
[1] 1 2 3 4 5

$z
[1] 1 0 1 0 1

$a
[1]  TRUE FALSE  TRUE FALSE  TRUE

$zf
[1] 1 0 1 0 1
Levels: 0 1
```

Unfortunately, having to know the various types of objects available in R may be discouraging or seem cumbersome to newcomers. Most readers can safely postpone a study of these things and learn these concepts only as needed. However, an understanding of classes of objects and related concepts described in this chapter will be seen to be helpful for readers wishing to exploit the true power of R.

HANDS-ON MATRIX ALGEBRA USING R - Active and Motivated Learning with Applications
© World Scientific Publishing Co. Pte. Ltd.
http://www.worldscibooks.com/mathematics/7814.html